



AXI Manager BFM Lite Version VIP – Lattice Propel Builder 2026.1.1

IP Version: v1.0.0

User Guide

FPGA-IPUG-02335-1.0

October 2026

Disclaimers

Lattice makes no warranty, representation, or guarantee regarding the accuracy of information contained in this document or the suitability of its products for any particular purpose. All information herein is provided AS IS, with all faults, and all associated risk is the responsibility entirely of the Buyer. The information provided herein is for informational purposes only and may contain technical inaccuracies or omissions, and may be otherwise rendered inaccurate for many reasons, and Lattice assumes no obligation to update or otherwise correct or revise this information. Products sold by Lattice have been subject to limited testing and it is the Buyer's responsibility to independently determine the suitability of any products and to test and verify the same. LATTICE PRODUCTS AND SERVICES ARE NOT DESIGNED, MANUFACTURED, OR TESTED FOR USE IN LIFE OR SAFETY CRITICAL SYSTEMS, HAZARDOUS ENVIRONMENTS, OR ANY OTHER ENVIRONMENTS REQUIRING FAIL-SAFE PERFORMANCE, INCLUDING ANY APPLICATION IN WHICH THE FAILURE OF THE PRODUCT OR SERVICE COULD LEAD TO DEATH, PERSONAL INJURY, SEVERE PROPERTY DAMAGE OR ENVIRONMENTAL HARM (COLLECTIVELY, "HIGH-RISK USES"). FURTHER, BUYER MUST TAKE PRUDENT STEPS TO PROTECT AGAINST PRODUCT AND SERVICE FAILURES, INCLUDING PROVIDING APPROPRIATE REDUNDANCIES, FAIL-SAFE FEATURES, AND/OR SHUT-DOWN MECHANISMS. LATTICE EXPRESSLY DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY OF FITNESS OF THE PRODUCTS OR SERVICES FOR HIGH-RISK USES. The information provided in this document is proprietary to Lattice Semiconductor, and Lattice reserves the right to make any changes to the information in this document or to any products at any time without notice.

Inclusive Language

This document was created consistent with Lattice Semiconductor's inclusive language policy. In some cases, the language in underlying tools and other items may not yet have been updated. Please refer to Lattice's inclusive language [FAQ 6878](#) for a cross reference of terms. Note in some cases such as register names and state names it has been necessary to continue to utilize older terminology for compatibility.

Contents

Contents	3
Abbreviations in This Document.....	5
1. Introduction	6
1.1. Features	6
1.2. Conventions	6
1.2.1. Nomenclature.....	6
2. Functional Descriptions	7
2.1. Overview	7
2.2. Signal Description.....	8
2.3. Attribute Summary.....	9
2.4. API Summary	9
3. AXI Manager BFM Lite Version VIP Generation.....	11
References	18
Technical Support Assistance	19
Revision History	20

Figures

Figure 2.1. AXI Manager BFM Lite Version VIP Block Diagram	7
Figure 3.1. Scalable RISC-V SoC Project	11
Figure 3.2. AXI Interconnect	11
Figure 3.3. AXI4 Feedthrough	12
Figure 3.4. Export of Feedthrough	12
Figure 3.5. Export of Clock and Reset	12
Figure 3.6. Switch Verification and SoC Design	13
Figure 3.7. AXI Manager BFM Lite Version VIP in IP Catalog	13
Figure 3.8. Entering Component Name	14
Figure 3.9. Configuring AXI Manager BFM Lite Version VIP	14
Figure 3.10. Verifying Result	15
Figure 3.11. Specifying Instance Name	15
Figure 3.12. Generated Instance	16
Figure 3.13. Connecting BFM Instance	16
Figure 3.14. Generating Testbench	16
Figure 3.15. Testbench Source File	17
Figure 3.16. Running Simulation	17
Figure 3.17. Simulation Output	17

Tables

Table 2.1. Clock and Reset Port	8
Table 2.2. AXI Manager Port	8
Table 2.3. Configurable Attributes	9
Table 2.4. API Summary	9

Abbreviations in This Document

A list of abbreviations used in this document.

Abbreviation	Definition
AXI	Advanced eXtensible Interface
API	Application Programming Interface
BFM	Bus Functional Model
DUT	Design Under Test
IP	Intellectual Property
RISC-V	Reduced Instruction Set Computer-V (Five)
RISC-V RX	RISC-V for RTOS applications
SoC	System-on-Chip
VIP	Verification IP

1. Introduction

The Lattice Semiconductor AXI Manager Bus Functional Model (BFM) Lite Version Verification IP (VIP) is for verification purposes only. It generates different types of AXI transactions, including single transactions, pipelined transactions, and burst transactions based on the AMBA4 AXI protocol.

The AXI Manager BFM Lite Version VIP design is implemented in SystemVerilog. It can be configured and generated using the Lattice Propel™ Builder software. It is for simulation only and is not targeted to any device.

1.1. Features

The AXI Manager BFM Lite Version VIP has the following features :

- AXI manager interface
- Single read and write transactions generation
- Pipelined read and write transactions generation
- INCR type burst read and write transactions generation

1.2. Conventions

1.2.1. Nomenclature

The nomenclature used in this document is based on SystemVerilog.

2. Functional Descriptions

2.1. Overview

The AXI Manager BFM Lite Version VIP generates AXI transactions for the verification environment, as shown in [Figure 2.1](#). Supported transactions include single read and write transactions, pipelined read and write transactions, and burst read and write transactions.

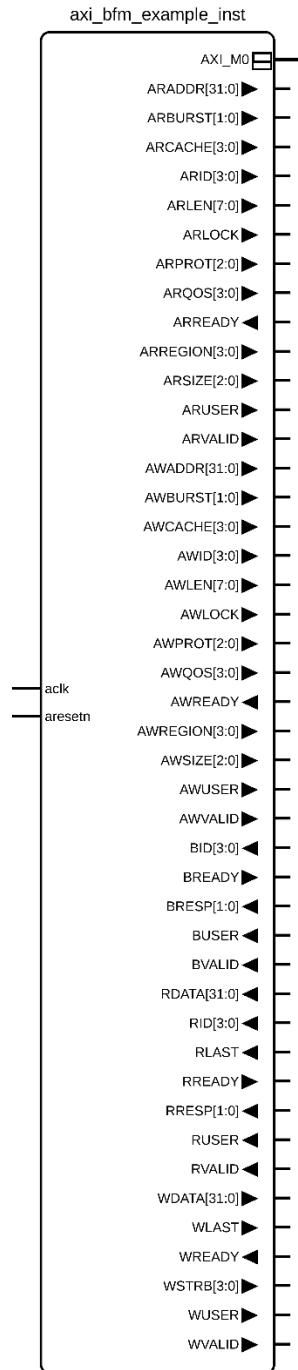


Figure 2.1. AXI Manager BFM Lite Version VIP Block Diagram

2.2. Signal Description

Table 2.1 and Table 2.2 list the ports of the AXI Manager BFM Lite Version VIP.

Table 2.1. Clock and Reset Port

Name	Type	Width	Description
aclk	in	1	Clock input
aresetn	in	1	Reset input, active low

Table 2.2. AXI Manager Port

Name	Type	Width	Description
ARADDR	out	ADDR_WIDTH	Standard AXI Manager Interface
ARBURST	out	2	Standard AXI Manager Interface
ARCACHE	out	4	Standard AXI Manager Interface
ARID	out	ID_WIDTH	Standard AXI Manager Interface
ARLEN	out	8	Standard AXI Manager Interface
ARLOCK	out	1	Standard AXI Manager Interface
ARPROT	out	3	Standard AXI Manager Interface
ARQOS	out	4	Standard AXI Manager Interface
ARREADY	in	1	Standard AXI Manager Interface
ARREGION	out	4	Standard AXI Manager Interface
ARSIZE	out	3	Standard AXI Manager Interface
ARUSER	out	USER_WIDTH	Standard AXI Manager Interface
ARVALID	out	1	Standard AXI Manager Interface
AWADDR	out	ADDR_WIDTH	Standard AXI Manager Interface
AWBURST	out	2	Standard AXI Manager Interface
AWCACHE	out	4	Standard AXI Manager Interface
AWID	out	ID_WIDTH	Standard AXI Manager Interface
AWLEN	out	8	Standard AXI Manager Interface
AWLOCK	out	1	Standard AXI Manager Interface
AWPROT	out	3	Standard AXI Manager Interface
AWQOS	out	4	Standard AXI Manager Interface
AWREADY	in	1	Standard AXI Manager Interface
AWREGION	out	4	Standard AXI Manager Interface
AWSIZE	out	3	Standard AXI Manager Interface
AWUSER	out	USER_WIDTH	Standard AXI Manager Interface
AWVALID	out	1	Standard AXI Manager Interface
BID	in	ID_WIDTH	Standard AXI Manager Interface
BREADY	out	1	Standard AXI Manager Interface
BRESP	in	2	Standard AXI Manager Interface
BUSER	in	USER_WIDTH	Standard AXI Manager Interface
BVALID	in	1	Standard AXI Manager Interface
RDATA	in	32	Standard AXI Manager Interface
RID	in	ID_WIDTH	Standard AXI Manager Interface
RLAST	in	1	Standard AXI Manager Interface
RREADY	out	1	Standard AXI Manager Interface
RRESP	in	2	Standard AXI Manager Interface
RUSER	in	USER_WIDTH	Standard AXI Manager Interface

Name	Type	Width	Description
RVALID	in	1	Standard AXI Manager Interface
WDATA	out	32	Standard AXI Manager Interface
WLAST	out	1	Standard AXI Manager Interface
WREADY	in	1	Standard AXI Manager Interface
WSTRB	out	4	Standard AXI Manager Interface
WUSER	out	USER_WIDTH	Standard AXI Manager Interface
WVALID	out	1	Standard AXI Manager Interface

Refer to the [AMBA® AXI and ACE Protocol Specification](#) for more information.

2.3. Attribute Summary

The configurable attributes of the AXI Manager BFM Lite Version VIP are shown in [Table 2.3](#). The attributes can be configured through the Lattice Propel Builder software.

Table 2.3. Configurable Attributes

Attribute	Selectable Values	Default	Description
Address Width	13–64	32	Address bus width
ID Width	2–11	4	ID signal width
User Width	1–128	1	User signal width

2.4. API Summary

Table 2.4. API Summary

API	Description
write_word_single (input logic [ADDR_WIDTH-1:0] addr, input logic [DATA_WIDTH-1:0] wdata)	Generate a single word write transaction. “addr” is the target address, and “wdata” is the word to be written.
write_word_pipeline (input logic [ADDR_WIDTH-1:0] addr[], input logic [DATA_WIDTH-1:0] wdata[])	Generate pipelined word write transactions. “addr[]” contains the target addresses, and “wdata[]” contains the words to be written.
write_word_burst_incr (input logic [ADDR_WIDTH-1:0] base_addr, input int beats, input logic [DATA_WIDTH-1:0] wdata[])	Generate an incremental burst write transaction. “base_addr” is the start address, “beats” is the burst beats and should be less than or equal to 256, and “wdata[]” contains the words to be written.
read_word_single (input logic [ADDR_WIDTH-1:0] addr, output logic [DATA_WIDTH-1:0] rdata)	Generate a single word read transaction. “addr” is the target address, and “rdata” is connected to RDATA and reflects the read data.
read_word_pipeline (input logic [ADDR_WIDTH-1:0] addr[], input int rnums, output logic [DATA_WIDTH-1:0] rdata[])	Generate pipelined word read transactions. “addr[]” contains the target addresses, “rnums” is the number of read operations, and “rdata[]” is connected to RDATA and reflects the read data.
read_word_burst_incr (input logic [ADDR_WIDTH-1:0] base_addr, input int beats, output logic [DATA_WIDTH-1:0] rdata[])	Generate an incremental burst read transaction. “base_addr” is the start address, “beats” is the burst beats and should be less than or equal to 256, and “rdata[]” is connected to RDATA and reflects the read data.

The following example shows the API usage.

```
logic [31:0] waddr_word_single = 32'h00000010;
logic [31:0] wdata_word_single = 32'h10101010;

logic [31:0] raddr_word_single = 32'h00000010;
logic [31:0] rdata_word_single;
```

```
logic [31:0] waddr_word_pipeline[3] = '{32'h00000014, 32'h00000018, 32'h0000001C};
logic [31:0] wdata_word_pipeline[3] = '{32'h20202020, 32'h30303030, 32'h40404040};

logic [31:0] raddr_word_pipeline[3] = '{32'h00000010, 32'h00000014, 32'h00000018};
int rnums_word_pipeline = 3;
logic [31:0] rdata_word_pipeline[3];

logic [31:0] waddr_word_burst_incr = 32'h00000010;
int wbeats_word_burst_incr = 5;
logic [31:0] wdata_word_burst_incr[5] = '{32'h00001001, 32'h00001002, 32'h00001003,
32'h00001004, 32'h00001005};

logic [31:0] raddr_word_burst_incr = 32'h00000010;
int rbeats_word_burst_incr = 5;
logic [31:0] rdata_word_burst_incr[5];

initial begin
    wait(clk_rst_inst_tb_rst_o_net);
    repeat(5) @(posedge clk_rst_inst_tb_clk_o_net);
    // Single write
    axi_mst_inst.write_word_single(waddr_word_single, wdata_word_single);
    // Single read
    axi_mst_inst.read_word_single(raddr_word_single, rdata_word_single);
    // Pipelined write
    axi_mst_inst.write_word_pipeline(waddr_word_pipeline, wdata_word_pipeline);
    // Pipelined read
    axi_mst_inst.read_word_pipeline(raddr_word_pipeline, rnums_word_pipeline,
rdata_word_pipeline);
    // INCR burst write
    axi_mst_inst.write_word_burst_incr(waddr_word_burst_incr, wbeats_word_burst_incr,
wdata_word_burst_incr);
    // INCR burst read
    axi_mst_inst.read_word_burst_incr(raddr_word_burst_incr, rbeats_word_burst_incr,
rdata_word_burst_incr);
end
```

3. AXI Manager BFM Lite Version VIP Generation

This section provides information on how to generate the AXI Manager BFM Lite Version VIP module using the Lattice Propel Builder software.

To generate the AXI Manager BFM Lite Version VIP module:

1. In the Lattice Propel Builder software, select **Scalable RISC-V SoC Project** and then select the RISC-V RX core (Figure 3.1).

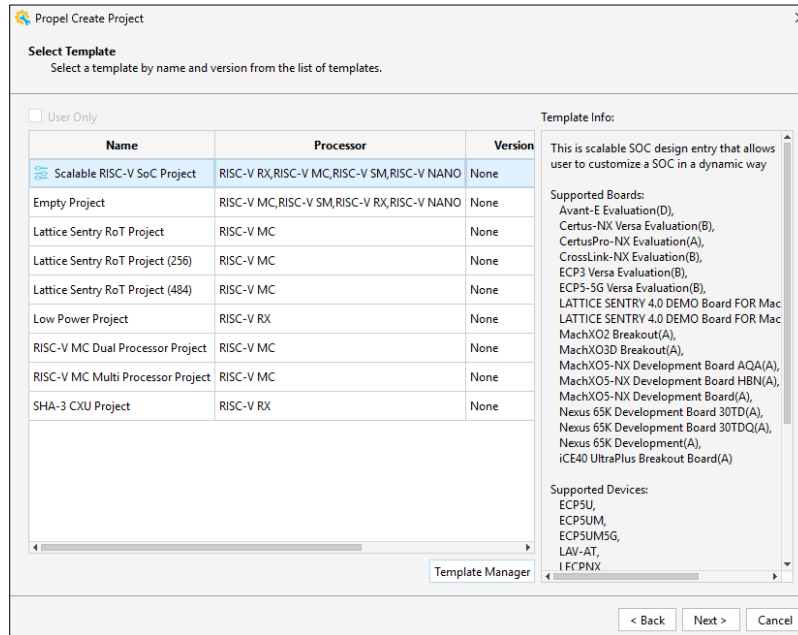


Figure 3.1. Scalable RISC-V SoC Project

2. Modify the AXI Interconnect instance and increase the Total External AXI Managers by one (Figure 3.2).

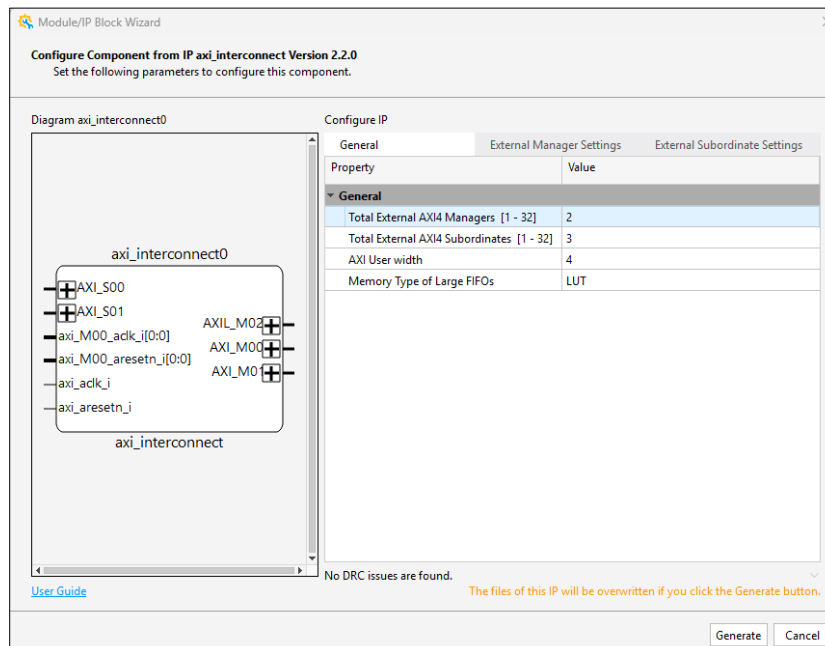


Figure 3.2. AXI Interconnect

3. Add the AXI4 Feedthrough IP to the design. Export the interface as Manager (Figure 3.3).

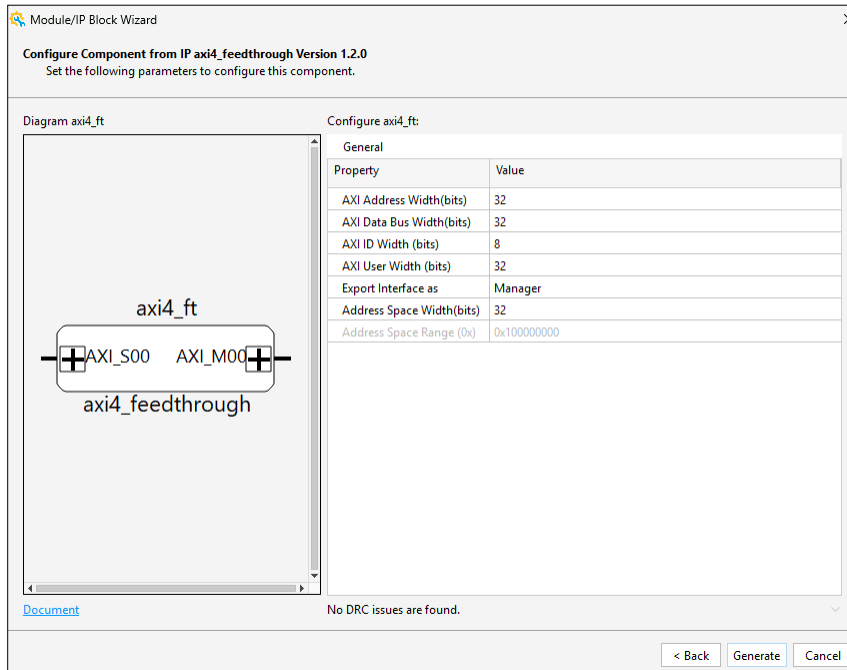


Figure 3.3. AXI4 Feedthrough

4. Connect the feedthrough instance to the AXI interconnect and export the interface to the top level (Figure 3.4).



Figure 3.4. Export of Feedthrough

5. Export the clock and reset to the top level (Figure 3.5).

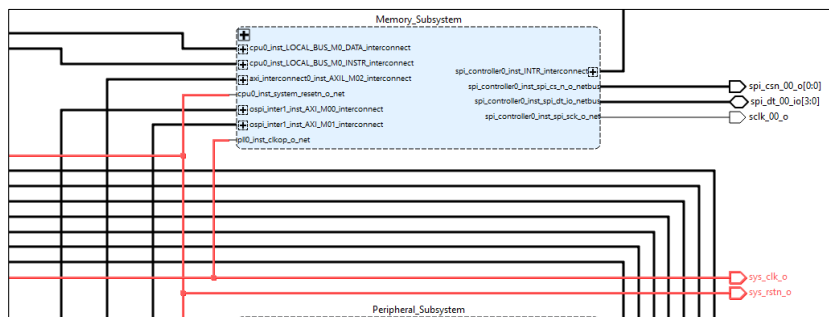


Figure 3.5. Export of Clock and Reset

6. Select **Design > Switch Verification and SoC Design** (Figure 3.6).

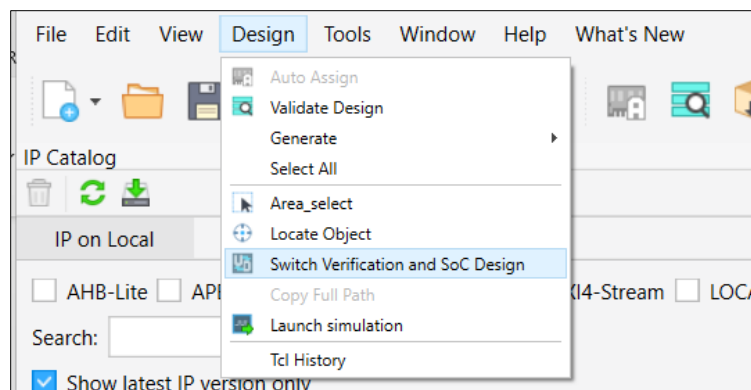


Figure 3.6. Switch Verification and SoC Design

7. Select the AXI Manager BFM Lite Version VIP from the IP Catalog (Figure 3.7).

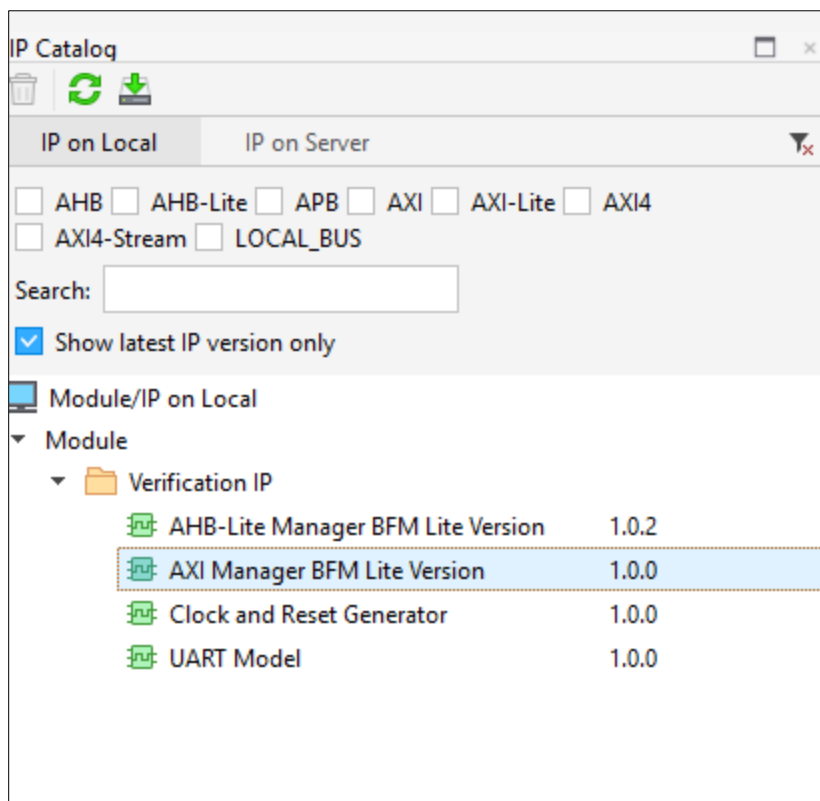


Figure 3.7. AXI Manager BFM Lite Version VIP in IP Catalog

8. Enter the component name. Click **Next** (Figure 3.8).

Module/IP Block Wizard

Generate Component from Module AXI_Manager_BFM_Lite Version 1.0.0
This wizard will guide you through the configuration, generation and instantiation of this Module/IP. Enter the following information to get started.

Component name:

Create in:

Next > **Cancel**

Figure 3.8. Entering Component Name

- Configure the attributes of the AXI Manager BFM Lite Version VIP as needed. Click **Generate** (Figure 3.9).

Module/IP Block Wizard

Configure Component from Module AXI_Manager_BFM_Lite Version 1.0.0
Set the following parameters to configure this component.

Diagram axi_bfm_example

axi_bfm_example

ack AXI_M0

aresetn

AXI_Manager_BFM_Lite

Configure IP

Property	Value
General	
Address Width [13 - 64]	32
ID Width [2 - 11]	8
User Width [1 - 128]	32

No DRC issues are found.

The files of this IP will be overwritten if you click the Generate button.

Generate **Cancel**

Figure 3.9. Configuring AXI Manager BFM Lite Version VIP

- Verify the information. Click **Finish** (Figure 3.10).

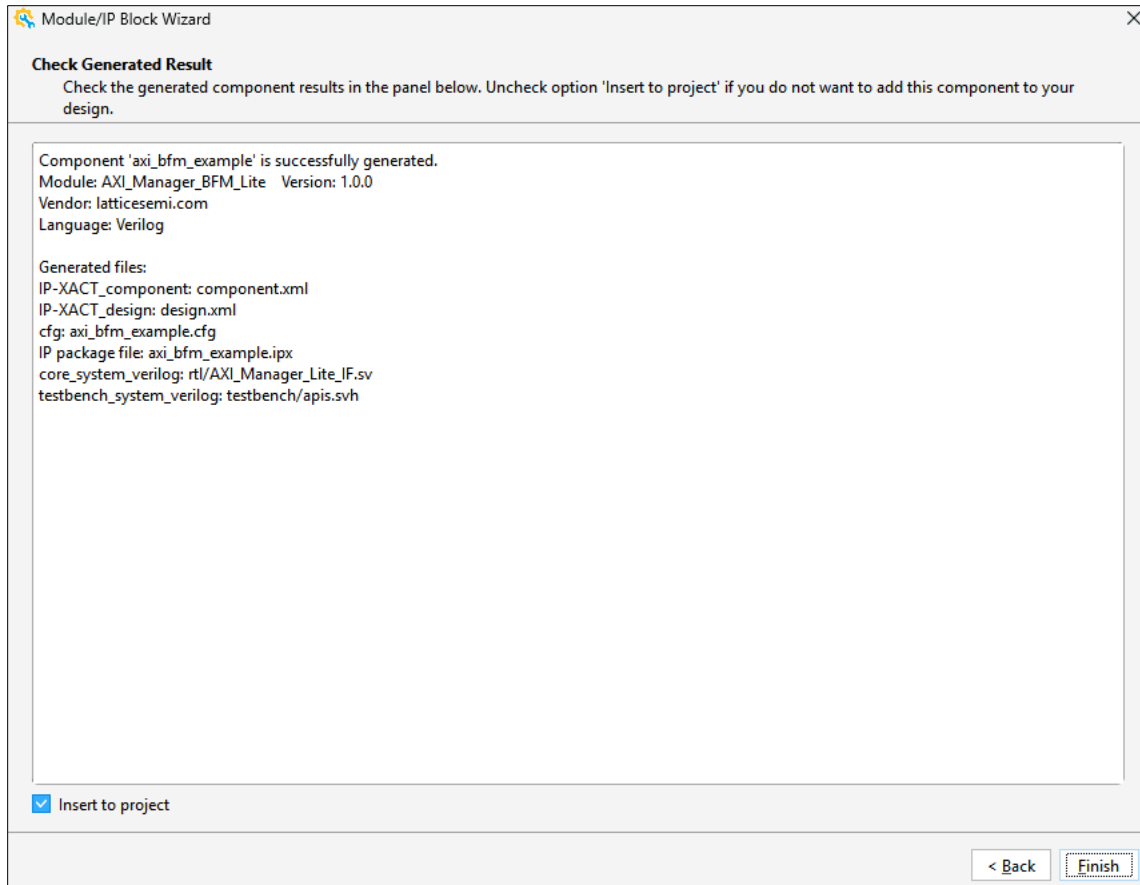


Figure 3.10. Verifying Result

11. Confirm or modify the module instance name. Click **OK** (Figure 3.11).

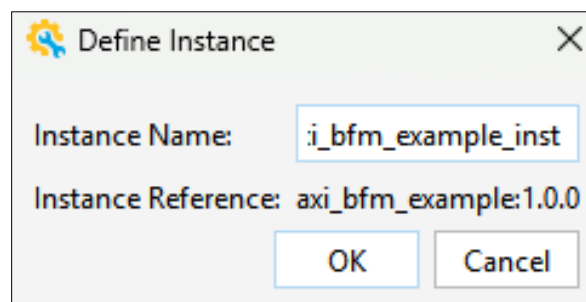


Figure 3.11. Specifying Instance Name

12. The AXI Manager BFM Lite Version VIP instance is successfully generated (Figure 3.12).

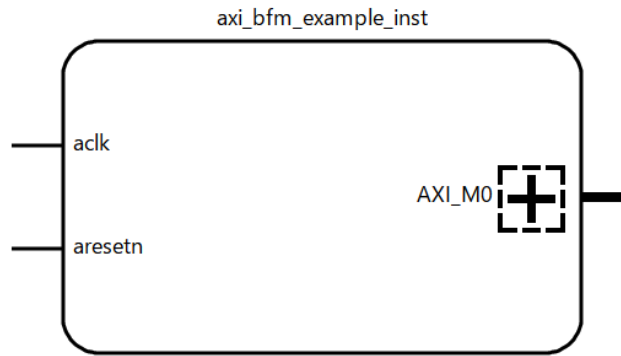


Figure 3.12. Generated Instance

13. Connect the AXI Manager BFM Lite Version VIP instance to the design under test (DUT) instance (Figure 3.13).

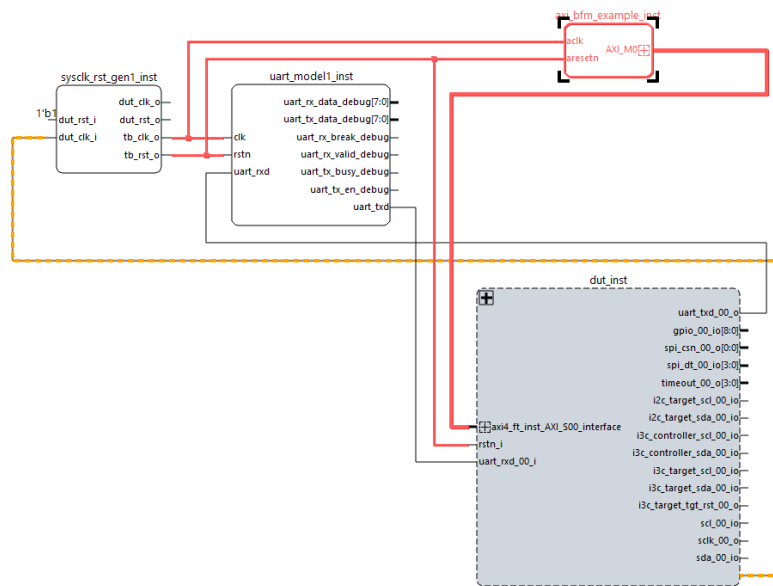


Figure 3.13. Connecting BFM Instance

14. Click the **Generate**  icon to generate the testbench. The Simulation Wizard window appears. Set the run time for simulation and click **Generate** (Figure 3.14).

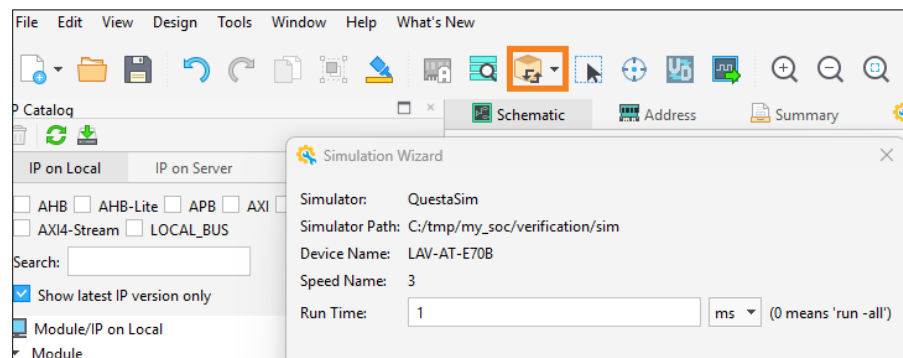


Figure 3.14. Generating Testbench

15. In a text editor, open the testbench source file. For example:
`<your design path>\<design>\verification\sim\<design>_v.sv.`
 Copy the example code of API usage from the [API Summary](#) section into the selected area shown in [Figure 3.15](#).

```

123
124      /*-----BFM Block-----*/
125      /* This section is reserved for user to create stimulus using BFM */
126      /* APIs. */
127      /*-----*/
128
129
130      initial begin
131          //Enabled Predefined sequences
132
133          //Reserved for User to write BFM
134
135      end
136
    
```

Figure 3.15. Testbench Source File

16. Re-run the simulation using either of the following options ([Figure 3.16](#)).
- Close the simulator and relaunch it.
 - Type `quit -sim` in the Questa Sim transcript area. Then, type `do qsim.do`.

```

# [1726000] AXI BFM: read_word_single A:0x30000 D:0x1
VSIM 4> quit -sim
# End time: 17:08:09 on Mar 26,2026, Elapsed time: 0:00:54
# Errors: 0, Warnings: 10

Questa Lattice OEM> do qsim.do
    
```

Figure 3.16. Running Simulation

17. The transaction is displayed, as shown in [Figure 3.17](#).

```

Testbench clock frequency set to 50.000000 Mhz, clock period set to 20.000000 ns
[63092000] AXI BFM: write_word_single A:0x10 D:0x10101010
[63402000] AXI BFM: read_word_single A:0x10 D:0x0
[63422000] AXI BFM: write_word_pipeline A:0x14 D:0x20202020
[63432000] AXI BFM: write_word_pipeline A:0x18 D:0x30303030
[63442000] AXI BFM: write_word_pipeline A:0x1c D:0x40404040
[63852000] AXI BFM: read_word_pipeline A:0x10 D:0x0
[63892000] AXI BFM: read_word_pipeline A:0x14 D:0x0
[63952000] AXI BFM: read_word_pipeline A:0x18 D:0x0
[63972000] AXI BFM: write_word_burst_incr beat 0 A:0x10 D:0x1001
[63982000] AXI BFM: write_word_burst_incr beat 1 A:0x14 D:0x1002
[63992000] AXI BFM: write_word_burst_incr beat 2 A:0x18 D:0x1003
[64002000] AXI BFM: write_word_burst_incr beat 3 A:0x1c D:0x1004
[64012000] AXI BFM: write_word_burst_incr beat 4 A:0x20 D:0x1005
[64282000] AXI BFM: read_word_burst_incr beat 0 A:0x10 D:0x0
[64302000] AXI BFM: read_word_burst_incr beat 1 A:0x14 D:0x0
[64312000] AXI BFM: read_word_burst_incr beat 2 A:0x18 D:0x0
[64322000] AXI BFM: read_word_burst_incr beat 3 A:0x1c D:0x0
[64332000] AXI BFM: read_word_burst_incr beat 4 A:0x20 D:0x0
    
```

Figure 3.17. Simulation Output

References

- [Lattice Propel Builder 2026.1 User Guide \(FPGA-UG-02254\)](#)
- [AMBA® AXI and ACE Protocol Specification](#)
- [Lattice Propel design environment web page](#)
- [Lattice Radiant™ software web page](#)
- [Lattice Diamond™ software web page](#)
- [Lattice Insights](#) for Lattice Semiconductor Training Series and Learning Plans

Technical Support Assistance

Submit a technical support case through www.latticesemi.com/techsupport.

For frequently asked questions, please refer to the Lattice Answer Database at www.latticesemi.com/Support/AnswerDatabase.

Revision History

Revision 1.0, IP v1.0.0, October 2026

Section	Change Summary
All	Production release.



www.latticesemi.com